Kit di strumenti per automazione DB

NetApp Solutions

NetApp
April 26, 2024

Sommario

Kit di strumenti per automazione DB

Automazione del ciclo di vita dei cloni Oracle di SnapCenter

Allen Cao, Niyaz Mohamed, NetApp

Scopo

I clienti apprezzano la funzionalità FlexClone dello storage NetApp ONTAP per i database che offre significativi risparmi sui costi di storage. Questo toolkit basato su Ansible automatizza setup, cloning e aggiornamento dei database Oracle clonati in base alle tempistiche, utilizzando le utilità della riga di comando di NetApp SnapCenter per una gestione ottimizzata del ciclo di vita. Il toolkit è applicabile ai database Oracle implementati sullo storage ONTAP on-premise o nel cloud pubblico e gestiti dal tool dell'interfaccia utente di NetApp SnapCenter.

Questa soluzione risolve i seguenti casi di utilizzo:

- Setup del file di configurazione delle specifiche dei cloni del database Oracle.
- Creare e aggiornare il database Oracle clone in base alla pianificazione definita dall'utente.

Pubblico

Questa soluzione è destinata alle seguenti persone:

- Un DBA che gestisce i database Oracle con SnapCenter.
- Un amministratore dello storage che gestisce lo storage ONTAP con SnapCenter.
- Proprietario di un'applicazione che ha accesso all'interfaccia utente di SnapCenter.

Licenza

Accedendo, scaricando, installando o utilizzando il contenuto di questo repository GitHub, l'utente accetta i termini della licenza riportata in "File di licenza".

> Ci sono alcune restrizioni riguardo alla produzione e/o alla condivisione di qualsiasi opera derivata con il contenuto di questo repository GitHub. Prima di utilizzare il contenuto, leggere i termini della licenza. Se non si accettano tutti i termini, non accedere, scaricare o utilizzare il contenuto di questo repository.

Implementazione della soluzione

Prerequisiti per l'implementazione

L’implementazione richiede i seguenti prerequisiti.

```
Ansible controller:
  Ansible v.2.10 and higher
  ONTAP collection 21.19.1
  Python 3
  Python libraries:
    netapp-lib
    xmltodict
    jmespath
```

```
SnapCenter server:
  version 5.0
  backup policy configured
  Source database protected with a backup policy
```

```
Oracle servers:
  Source server managed by SnapCenter
  Target server managed by SnapCenter
  Target server with identical Oracle software stack as source server
installed and configured
```

Scaricare il toolkit

```
git clone https://bitbucket.ngage.netapp.com/scm/ns-
bb/na_oracle_clone_lifecycle.git
```

Configurazione dei file host di destinazione Ansible

Il toolkit include un file hosts che definisce le destinazioni per cui viene eseguito un playbook Ansible. In genere, si tratta degli host clone di Oracle di destinazione. Di seguito è riportato un file di esempio. Una voce dell'host include l'indirizzo IP dell'host di destinazione e la chiave ssh per l'accesso di un utente amministratore all'host per eseguire il comando clone o refresh.

#Host cloni Oracle

```
[clone_1]
ora_04.cie.netapp.com ansible_host=10.61.180.29
ansible_ssh_private_key_file=ora_04.pem
```

```
[clone_2]
[clone_3]
```

Configurazione variabili globali

I playbook Ansible prendono input variabili da diversi file variabili. Di seguito è riportato un esempio di file variabile globale vars.yml.

```
# ONTAP specific config variables
# SnapCtr specific config variables
```

```
snapctr_usr: xxxxxxxx
snapctr_pwd: 'xxxxxxxx'
```

```
backup_policy: 'Oracle Full offline Backup'
# Linux specific config variables
# Oracle specific config variables
```

Configurazione variabili host

Le variabili host sono definite nella directory host_vars denominata {{ host_name }}.yml. Di seguito è riportato un esempio di file di variabile host Oracle di destinazione ora_04.cie.netapp.com.yml che mostra la configurazione tipica.

```
# User configurable Oracle clone db host specific parameters
```

```
# Source database to clone from
source_db_sid: NTAP1
source_db_host: ora_03.cie.netapp.com
```

```
# Clone database
clone_db_sid: NTAP1DEV
```

```
snapctr_obj_id: '{{ source_db_host }}\{{ source_db_sid }}'
```

Configurazione aggiuntiva del server Oracle di destinazione dei cloni

Il server Oracle di destinazione della clonazione deve avere lo stesso stack software Oracle del server Oracle di origine installato e sottoposto a patch. L'utente Oracle .bash_profile ha $ORACLE_BASE e $ORACLE_HOME configurato. Inoltre, la variabile $ORACLE_HOME deve corrispondere all'impostazione del server Oracle di origine. Di seguito viene riportato un esempio.

```
# .bash_profile
```

```
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
```

```
# User specific environment and startup programs
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=/u01/app/oracle/product/19.0.0/NTAP1
```

Esecuzione Playbook

Sono disponibili un totale di tre playbook per eseguire il ciclo di vita dei cloni del database Oracle con le utility della CLI di SnapCenter.

1. Installare i prerequisiti del controller Ansible - una sola volta.

```
ansible-playbook -i hosts ansible_requirements.yml
```

2. File di configurazione clone - una sola volta.

```
ansible-playbook -i hosts clone_1_setup.yml -u admin -e
@vars/vars.yml
```

3. Crea e aggiorna regolarmente il database dei cloni da crontab con uno script shell per chiamare un playbook di refresh.

```
0 */4 * * * /home/admin/na_oracle_clone_lifecycle/clone_1_refresh.sh
```

Per un database clone aggiuntivo, creare un clone_n_setup.yml e clone_n_refresh.yml separati e clone_n_refresh.sh. Configurare di conseguenza gli host di destinazione Ansible e il file hostname.yml nella directory host_vars.

Dove trovare ulteriori informazioni

Per ulteriori informazioni sull'automazione delle soluzioni NetApp, consulta il seguente sito Web "Automazione delle soluzioni NetApp"

Migrazione Oracle automatizzata

Team di progettazione delle soluzioni NetApp

Scopo

Questo toolkit automatizza la migrazione del database Oracle da on-premise al cloud AWS con storage FSX ONTAP e istanza di calcolo EC2 come infrastruttura di destinazione. Si presuppone che il cliente disponga già di un database Oracle on-premise implementato nel modello CDB/PDB. Il toolkit consentirà al cliente di trasferire un PDB denominato da un database di container su un host Oracle utilizzando la procedura di trasferimento di Oracle PDB con un'opzione di massima disponibilità. Ciò significa che il PDB di origine su qualsiasi storage array on-premise viene ricollocato in un nuovo database dei container, con un'interruzione minima del servizio. La procedura di trasferimento di Oracle sposterà i file di dati Oracle mentre il database è online. Successivamente, esegue il reindirizzamento delle sessioni utente dai servizi di database on-premise ai servizi di database ricollocati al momento del trasferimento, quando tutti i file di dati passano nel cloud di AWS. La tecnologia sottolineata è la metodologia collaudata per i cloni a caldo dei database Oracle PDB.

> Sebbene il toolkit di migrazione sia sviluppato e validato sull'infrastruttura cloud di AWS, si basa sulle soluzioni Oracle a livello di applicazione. Pertanto, il toolkit è applicabile ad altre piattaforme di cloud pubblico, come Azure, GCP, ecc.

Questa soluzione risolve i seguenti casi di utilizzo:

- Creare un utente di migrazione e concedere i privilegi richiesti sul server DB di origine on-premise.
- Sposta un PDB da CDB on-premise a un CDB di destinazione nel cloud mentre il PDB di origine è online fino al momento del passaggio.

Pubblico

Questa soluzione è destinata alle seguenti persone:

- Un DBA che migra i database Oracle da on-premise al cloud AWS.
- Un Solution Architect per database interessato alla migrazione dei database Oracle da risorse on-premise al cloud AWS.
- Un amministratore dello storage che gestisce lo storage AWS FSX ONTAP con supporto per i database Oracle.
- Un proprietario delle applicazioni che ama migrare i database Oracle da storage on-premise al cloud AWS.

Licenza

Accedendo, scaricando, installando o utilizzando il contenuto di questo repository GitHub, l'utente accetta i termini della licenza riportata in "File di licenza".

> Ci sono alcune restrizioni riguardo alla produzione e/o alla condivisione di qualsiasi opera derivata con il contenuto di questo repository GitHub. Prima di utilizzare il contenuto, leggere i termini della licenza. Se non si accettano tutti i termini, non accedere, scaricare o utilizzare il contenuto di questo repository.

Implementazione della soluzione

Prerequisiti per l'implementazione

L’implementazione richiede i seguenti prerequisiti.

```
Ansible v.2.10 and higher
ONTAP collection 21.19.1
Python 3
Python libraries:
  netapp-lib
  xmltodict
  jmespath
```

```
Source Oracle CDB with PDBs on-premises
Target Oracle CDB in AWS hosted on FSx and EC2 instance
Source and target CDB on same version and with same options installed
```

```
Network connectivity
  Ansible controller to source CDB
  Ansible controller to target CDB
  Source CDB to target CDB on Oracle listener port (typical 1521)
```

Scaricare il toolkit

```
git clone https://github.com/NetApp/na_ora_aws_migration.git
```

Configurazione variabili host

Le variabili host sono definite nella directory host_vars denominata {{ host_name }}.yml. Un esempio di file di variabile host host_name.yml è incluso per dimostrare la configurazione tipica. Di seguito sono riportate alcune considerazioni fondamentali:

```
Source Oracle CDB - define host specific variables for the on-prem CDB
  ansible_host: IP address of source database server host
  source_oracle_sid: source Oracle CDB instance ID
  source_pdb_name: source PDB name to migrate to cloud
  source_file_directory: file directory of source PDB data files
  target_file_directory: file directory of migrated PDB data files
```

```
Target Oracle CDB - define host specific variables for the target CDB
including some variables for on-prem CDB
  ansible_host: IP address of target database server host
  target_oracle_sid: target Oracle CDB instance ID
  target_pdb_name: target PDB name to be migrated to cloud (for max
availability option, the source and target PDB name must be the same)
  source_oracle_sid: source Oracle CDB instance ID
  source_pdb_name: source PDB name to be migrated to cloud
  source_port: source Oracle CDB listener port
  source_oracle_domain: source Oracle database domain name
  source_file_directory: file directory of source PDB data files
  target_file_directory: file directory of migrated PDB data files
```

Configurazione del file host del server DB

L'istanza di AWS EC2 utilizza l'indirizzo IP per la denominazione dell'host per impostazione predefinita. Se usi un nome diverso nel file hosts per Ansible, configura la risoluzione dei nomi degli host nel file /etc/hosts per il server di origine e di destinazione. Di seguito viene riportato un esempio.

```
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
172.30.15.96 source_db_server
172.30.15.107 target_db_server
```

Esecuzione Playbook - eseguita in sequenza

1. Installare i prerequisiti del controller Ansible.

```
ansible-playbook -i hosts requirements.yml
```

```
ansible-galaxy collection install -r collections/requirements.yml
--force
```

2. Eseguire attività di pre-migrazione su server on-premise, supponendo che l'amministratore sia un utente ssh per la connessione all'host Oracle on-premise con l'autorizzazione sudo.

```
ansible-playbook -i hosts ora_pdb_relocate.yml -u admin -k -K -t
ora_pdb_relo_onprem
```

3. Esegui il trasferimento di Oracle PDB dal CDB on-premise al CDB di destinazione nell'istanza di AWS EC2, supponendo che EC2 utente per una connessione all'istanza del DB di EC2 MB e DB1.pem con coppie di chiavi ssh di EC2 utenti.

```
ansible-playbook -i hosts ora_pdb_relocate.yml -u ec2-user --private
-key db1.pem -t ora_pdb_relo_primary
```

Dove trovare ulteriori informazioni

Per ulteriori informazioni sull'automazione delle soluzioni NetApp, consulta il seguente sito Web "Automazione delle soluzioni NetApp"

Automazione di ha/DR Oracle in AWS FSX ONTAP

Team di progettazione delle soluzioni NetApp

Scopo

Questo toolkit automatizza le attività di configurazione e gestione di un ambiente HR/DR (High Availability and Disaster Recovery) per database Oracle implementati nel cloud AWS con FSX per lo storage ONTAP e le istanze di calcolo EC2.

Questa soluzione risolve i seguenti casi di utilizzo:

- Configurazione host di destinazione ha/DR - configurazione del kernel, configurazione di Oracle che corrisponda all'host del server di origine.
- Setup FSX ONTAP - peering dei cluster, peering dei vserver, configurazione delle relazioni di Oracle Volumes snapmirror dall'origine alla destinazione.
- Backup dei dati del database Oracle tramite snapshot - esecuzione fuori crontab

- Backup del log di archivio del database Oracle tramite snapshot - esecuzione fuori crontab
- Esecuzione di failover e recovery sull'host ha/DR: Test e convalida dell'ambiente ha/DR
- Esegui la risincronizzazione dopo il test di failover - ristabilire la relazione di snapmirror dei volumi di database in modalità ha/DR

Pubblico

Questa soluzione è destinata alle seguenti persone:

- Un DBA che ha configurato il database Oracle in AWS per ottenere alta disponibilità, protezione dei dati e disaster recovery.
- Un Solution architect per database interessato a una soluzione ha/DR Oracle a livello di storage nel cloud AWS.
- Un amministratore dello storage che gestisce lo storage AWS FSX ONTAP con supporto per i database Oracle.
- Un proprietario di applicazioni che desidera supportare un database Oracle per ha/DR nell'ambiente AWS FSX/EC2.

Licenza

Accedendo, scaricando, installando o utilizzando il contenuto di questo repository GitHub, l'utente accetta i termini della licenza riportata in "File di licenza".

> Ci sono alcune restrizioni riguardo alla produzione e/o alla condivisione di qualsiasi opera derivata con il contenuto di questo repository GitHub. Prima di utilizzare il contenuto, leggere i termini della licenza. Se non si accettano tutti i termini, non accedere, scaricare o utilizzare il contenuto di questo repository.

Implementazione della soluzione

Prerequisiti per l'implementazione

L’implementazione richiede i seguenti prerequisiti.

```
Ansible v.2.10 and higher
ONTAP collection 21.19.1
Python 3
Python libraries:
  netapp-lib
  xmltodict
  jmespath
```

```
AWS FSx storage as is available
```

```
AWS EC2 Instance
  RHEL 7/8, Oracle Linux 7/8
  Network interfaces for NFS, public (internet) and optional management
  Existing Oracle environment on source, and the equivalent Linux
operating system at the target
```

Scaricare il toolkit

```
git clone https://github.com/NetApp/na_ora_hadr_failover_resync.git
```

Configurazione variabili globali

I playbook Ansible sono basati su variabili. Un esempio di file variabile globale fsx_vars_example.yml è incluso per dimostrare la configurazione tipica. Di seguito sono riportate alcune considerazioni fondamentali:

```
ONTAP - retrieve FSx storage parameters using AWS FSx console for both
source and target FSx clusters.
  cluster name: source/destination
  cluster management IP: source/destination
  inter-cluster IP: source/destination
  vserver name: source/destination
  vserver management IP: source/destination
  NFS lifs: source/destination
  cluster credentials: fsxadmin and vsadmin pwd to be updated in
roles/ontap_setup/defaults/main.yml file
```

```
Oracle database volumes - they should have been created from AWS FSx
console, volume naming should follow strictly with following standard:
  Oracle binary: {{ host_name }}_bin, generally one lun/volume
  Oracle data: {{ host_name }}_data, can be multiple luns/volume, add
additional line for each additional lun/volume in variable such as {{
host_name }}_data_01, {{ host_name }}_data_02 ...
  Oracle log: {{ host_name }}_log, can be multiple luns/volume, add
additional line for each additional lun/volume in variable such as {{
host_name }}_log_01, {{ host_name }}_log_02 ...
  host_name: as defined in hosts file in root directory, the code is
written to be specifically matched up with host name defined in host
file.
```

```
Linux and DB specific global variables - keep it as is.
  Enter redhat subscription if you have one, otherwise leave it black.
```

Configurazione variabili host

Le variabili host sono definite nella directory host_vars denominata {{ host_name }}.yml. Un esempio di file di variabile host host_name.yml è incluso per dimostrare la configurazione tipica. Di seguito sono riportate alcune considerazioni fondamentali:

```
Oracle - define host specific variables when deploying Oracle in
multiple hosts concurrently
  ansible_host: IP address of database server host
  log_archive_mode: enable archive log archiving (true) or not (false)
  oracle_sid: Oracle instance identifier
  pdb: Oracle in a container configuration, name pdb_name string and
number of pdbs (Oracle allows 3 pdbs free of multitenant license fee)
  listener_port: Oracle listener port, default 1521
  memory_limit: set Oracle SGA size, normally up to 75% RAM
  host_datastores_nfs: combining of all Oracle volumes (binary, data,
and log) as defined in global vars file. If multi luns/volumes, keep
exactly the same number of luns/volumes in host_var file
```

```
Linux - define host specific variables at Linux level
  hugepages_nr: set hugepage for large DB with large SGA for
performance
  swap_blocks: add swap space to EC2 instance. If swap exist, it will
be ignored.
```

Configurazione del file host del server DB

L'istanza di AWS EC2 utilizza l'indirizzo IP per la denominazione dell'host per impostazione predefinita. Se usi un nome diverso nel file hosts per Ansible, configura la risoluzione dei nomi degli host nel file /etc/hosts per i server di origine e di destinazione. Di seguito viene riportato un esempio.

```
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
172.30.15.96 db1
172.30.15.107 db2
```

Esecuzione Playbook - eseguita in sequenza

1. Installa i prerequisiti del controller Ansible.

```
ansible-playbook -i hosts requirements.yml
```

```
ansible-galaxy collection install -r collections/requirements.yml
--force
```

2. Installare l'istanza del database EC2 di destinazione.

```
ansible-playbook -i hosts ora_dr_setup.yml -u ec2-user --private-key
db2.pem -e @vars/fsx_vars.yml
```

3. Configura la relazione di snapmirror di FSX ONTAP tra i volumi del database di origine e di destinazione.

```
ansible-playbook -i hosts ontap_setup.yml -u ec2-user --private-key
db2.pem -e @vars/fsx_vars.yml
```

4. Eseguire il backup dei volumi dei dati dei database Oracle tramite snapshot da crontab.

```
10 * * * * cd /home/admin/na_ora_hadr_failover_resync &&
/usr/bin/ansible-playbook -i hosts ora_replication_cg.yml -u ec2-
user --private-key db1.pem -e @vars/fsx_vars.yml >>
logs/snap_data_`date +"%Y-%m%d-%H%M%S"`.log 2>&1
```

5. Eseguire il backup dei volumi del registro di archivio dei database Oracle tramite snapshot da crontab.

```
0,20,30,40,50 * * * * cd /home/admin/na_ora_hadr_failover_resync &&
/usr/bin/ansible-playbook -i hosts ora_replication_logs.yml -u ec2-
user --private-key db1.pem -e @vars/fsx_vars.yml >>
logs/snap_log_`date +"%Y-%m%d-%H%M%S"`.log 2>&1
```

6. Esecuzione di failover e ripristino del database Oracle sull'istanza EC2 DB di destinazione per testare e convalidare la configurazione ha/DR.

```
ansible-playbook -i hosts ora_recovery.yml -u ec2-user --private-key
db2.pem -e @vars/fsx_vars.yml
```

7. Esegui la risincronizzazione dopo il test di failover - ristabilire la relazione di snapmirror dei volumi di database in modalità di replica.

```
ansible-playbook -i hosts ontap_ora_resync.yml -u ec2-user --private
-key db2.pem -e @vars/fsx_vars.yml
```

Dove trovare ulteriori informazioni

Per ulteriori informazioni sull'automazione delle soluzioni NetApp, consulta il seguente sito Web "Automazione delle soluzioni NetApp"

Cluster ONTAP AWS FSX e provisioning di istanze EC2

Team di progettazione delle soluzioni NetApp

Scopo

Questo toolkit automatizza le attività di provisioning di un cluster di storage AWS FSX ONTAP e di un'istanza di calcolo EC2, che può essere successivamente utilizzata per l'implementazione del database.

Questa soluzione risolve i seguenti casi di utilizzo:

- Esegui il provisioning di un'istanza di calcolo EC2 nel cloud AWS in una subnet VPC predefinita e imposta la chiave ssh per l'accesso a EC2 istanza come EC2 utente.
- Esegui il provisioning di un cluster di storage AWS FSX ONTAP nelle zone di disponibilità desiderate e configura una SVM di storage e imposta la password fsxadmin dell'utente del cluster.

Pubblico

Questa soluzione è destinata alle seguenti persone:

- Un DBA che gestisce i database in un ambiente AWS EC2.
- Un Solution Architect per database interessato all'implementazione dei database nell'ecosistema AWS EC2.
- Un amministratore dello storage che gestisce uno storage AWS FSX ONTAP che supporta i database.
- Un proprietario delle applicazioni che ama gestire il database nell'ecosistema AWS EC2.

Licenza

Accedendo, scaricando, installando o utilizzando il contenuto di questo repository GitHub, l'utente accetta i termini della licenza riportata in "File di licenza".

> Ci sono alcune restrizioni riguardo alla produzione e/o alla condivisione di qualsiasi opera derivata con il contenuto di questo repository GitHub. Prima di utilizzare il contenuto, leggere i termini della licenza. Se non si accettano tutti i termini, non accedere, scaricare o utilizzare il contenuto di questo repository.

Implementazione della soluzione

Prerequisiti per l'implementazione

L'implementazione richiede i seguenti prerequisiti.

```
An Organization and AWS account has been setup in AWS public cloud
  An user to run the deployment has been created
  IAM roles has been configured
  IAM roles granted to user to permit provisioning the resources
```

```
VPC and security configuration
  A VPC has been created to host the resources to be provisioned
  A security group has been configured for the VPC
  A ssh key pair has been created for EC2 instance access
```

```
Network configuration
  Subnets has been created for VPC with network segments assigned
  Route tables and network ACL configured
  NAT gateways or internet gateways configured for internet access
```

Scaricare il toolkit

```
git clone https://github.com/NetApp/na_aws_fsx_ec2_deploy.git
```

Connettività e autenticazione

Il toolkit deve essere eseguito da una shell del cloud AWS. La shell cloud di AWS è una shell basata sul browser che facilita la gestione, l'esplorazione e l'interazione in sicurezza con le tue risorse AWS. CloudShell è pre-autenticato con le credenziali della console dell'utente. Gli strumenti operativi e di sviluppo più comuni sono preinstallati, pertanto non è necessaria alcuna installazione o configurazione locale.

Configurazione dei file terraform provider.tf e main.tf

Il provider.tf definisce il provider dal quale Terraform effettua il provisioning delle risorse tramite chiamate API. Il file main.tf definisce le risorse e gli attributi delle risorse da sottoporre a provisioning. Di seguito sono riportati alcuni dettagli:

```
provider.tf:
  terraform {
    required_providers {
      aws = {
        source  = "hashicorp/aws"
        version = "~> 4.54.0"
      }
    }
  }
```

```
main.tf:
  resource "aws_instance" "ora_01" {
    ami                           = var.ami
    instance_type                 = var.instance_type
    subnet_id                     = var.subnet_id
    key_name                      = var.ssh_key_name
    root_block_device {
      volume_type                 = "gp3"
      volume_size                 = var.root_volume_size
    }
    tags = {
      Name                        = var.ec2_tag
    }
  }
  ....
```

Configurazione delle variabili di terraform.tf e terraform.tfvars

Variables.tf dichiara le variabili da utilizzare in main.tf. Il file terraform.tfvars contiene i valori effettivi per le variabili. Di seguito sono riportati alcuni esempi:

```
variables.tf:
  ### EC2 instance variables ###
```

```
variable "ami" {
  type        = string
  description = "EC2 AMI image to be deployed"
}
```

```
variable "instance_type" {
  type        = string
  description = "EC2 instance type"
}
```

```

```

terraform.tfvars:
EC2 instance variables

```

```

ami = "ami-06640050dc3f556bb" //RedHat 8.6 AMI
instance_type = "t2.micro"
ec2_tag = "ora_01"
subnet_id = "subnet-04f5fe7073ff514fb"
ssh_key_name = "sufi_new"
root_volume_size = 30

```

```

Procedure passo passo - eseguite in sequenza

1. Installa Terraform nella shell del cloud AWS.

```
git clone https://github.com/tfutils/tfenv.git ~/.tfenv
```

```
mkdir ~/bin
```

```
ln -s ~/.tfenv/bin/* ~/bin/
```

```
tfenv install
```

```
tfenv use 1.3.9
```

2. Scarica il toolkit dal sito pubblico di NetApp GitHub

```
git clone https://github.com/NetApp-
Automation/na_aws_fsx_ec2_deploy.git
```

3. Eseguire init per inizializzare la terraform

```
terraform init
```

4. Generare il piano di esecuzione

```
terraform plan -out=main.plan
```

5. Applicare il piano di esecuzione

```
terraform apply "main.plan"
```

6. Eseguire Destroy per rimuovere le risorse al termine dell'operazione

```
terraform destroy
```

Dove trovare ulteriori informazioni

Per ulteriori informazioni sull’automazione delle soluzioni NetApp, consulta il seguente sito Web "Automazione delle soluzioni NetApp"